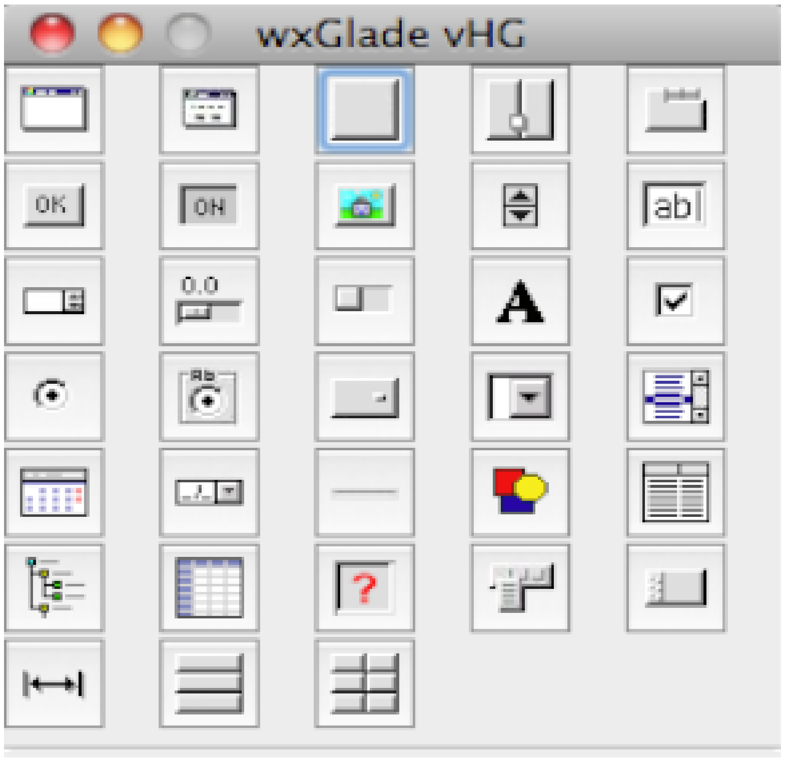

wxGlue Documentation

Release 0.4

Brian Toby

January 08, 2013

CONTENTS

1 *1. Introduction* **3**

2 *2. What is Implemented* **5**

3 *3. How to Use wxGlue* **7**

4 *4. Designing a wxGlue Program* **9**

4.1 Imports . . . 9
4.2 Define Function . . . 9
4.3 Define Callbacks . . . 9
4.4 Control Initalization . . . 10
4.5 Resize and Show the Frame . . . 10
4.6 Start the code . . . 10

5 *5. wxGlue Methods for Specific Controls* **13**

5.1 wx.TextCtrl methods . . . 13
5.2 wx.Button methods . . . 13
5.3 Calendar Methods . . . 13
5.4 Option Selection Methods . . . 14
5.5 StatusBar Methods . . . 14
5.6 ToolBar methods . . . 14
5.7 MenuBar methods . . . 14
5.8 Using a checkbox to enable other controls . . . 15

6 *6. Other Important Functions and Methods* **17**

7 *7. Reference: Class and Function Documentation* **19**

Python Module Index **27**

Index **29**

wxGlue Info

Release 0.4
Date January 08, 2013
Author **Brian H. Toby**
Target Novice Python programmers
Project Status in early development

Contents

- wxGlue: Classes to implement GUIs from wxGlade
  - *1. Introduction*
  - *2. What is Implemented*
  - *3. How to Use wxGlue*
  - *4. Designing a wxGlue Program*
    - Imports
    - Define Function
    - Define Callbacks
    - Control Initalization
    - Resize and Show the Frame
    - Start the code
  - *5. wxGlue Methods for Specific Controls*
    - wx.TextCtrl methods
    - wx.Button methods
    - Calendar Methods
    - Option Selection Methods
    - StatusBar Methods
    - ToolBar methods
    - MenuBar methods
    - Using a checkbox to enable other controls
  - *6. Other Important Functions and Methods*
  - *7. Reference: Class and Function Documentation*

CHAPTER

ONE

1. INTRODUCTION

This script contains a set of classes used to manage graphical user interface (GUI) controls in wxPython in a way that allows GUIs to be programmed without directly writing any wxPython calls. The GUI script is created using wxGlade (http://wxglade.sourceforge.net/) and then wxGlue takes over by providing a set of methods that interact with the GUI controls, such as loading settings into variables, validating inputs and responding to button presses.

To help in preparing the script that will use the wxGlue methods and functions, the wxGlue script also provides a function that will scan a script created with wxGlade and create a template script that has example code. The template can then be edited to better control the GUI and to introduce the functionality is desired.

Note that what most of the english speaking world calls a window (the box-like thing on the computer screen with little buttons and a title on the border) is called a frame in wxPython. The various things that one has inside a window, such as buttons, labels, etc. are called widgets (or confusingly windows) in wxPython. Here, they are called controls. The control inside a frame that looks like an empty rectangle, where one types information, is called a TextCtrl in wxPython.

CHAPTER

TWO

2. WHAT IS IMPLEMENTED

This package has only been tested with wxGlade. In theory one should be able to use other GUI builders such as BoaConstructer, but this has not been tested. The top-level widget should be a wx.Frame instance and the following wxPython controls are currently supported:

TextCtrl, Button, BitmapButton, StaticBitmap, DatePickerCtrl, CalendarCtrl, ToggleButton, SpinCtrl, SpinButton, CheckBox, RadioButton, Gauge, Slider, StaticText, StaticBox, RadioBox, Choice, ComboBox, ListBox, StatusBar, ToolBar, MenuBar

This includes most of the capabilities in wxGlade, except for wx.ListCtrl, wx.TreeCtrl, and wx.grid.Grid, Applications must have a wx.Frame or or wx.Dialog base class, but at least for now, frames using base class wx.MDIChildFrame are not supported. The following items have not been tested, but probably work: Notebooks, Panels/ScrolledWindows, SplitterWindow.

CHAPTER

THREE

3. HOW TO USE WXGLUE

The first step in using this package is to create a frame with a GUI using wxGlade.

1. Start wxGlade.
2. In the wxGlade Properties window, edit the application properties as needed. Be sure to set an output path: this should contain the entire name of the python file to be created. Note that wxGlue has been tested with settings: US-ASCII; no gettext support; Code generation: Single File; Language: Python (of course); wxWidgets: 2.8.
3. In the main wxGlade window click on the "Frame" button (upper left). A new window pops up, select the base class as wxFrame and name the class as you will. Click OK.
4. Note that a frame and a sizer has been added to the the wxGlade: Tree window. You will likely want to split this into more than one vertical section. Do this by right-clicking on the sizer entry in that window (Mac: control+click) and select the Add slot menu item as many times as needed.
5. To break a sizer section into multiple horizontally divided sections, click on the BoxSizer button in the main wxGlade window (near end). Then click in the Design window where this sizer will be placed. Select the Orientation: Horizontal, set the number of slots, as desired (usually 2 or more). Note that the Static Box option allows a label to be added to the border of the sizer.
6. Add controls to each section of the sizers, by clicking on the control icon in the main wxGlade window and then clicking in the design window where the widget will be placed. It is necessary to click on the on the control icon each time it a control will be added, even if it is a repeat of the previous.
7. Edit parameters for a control by changing information in the properties window. If needed, a control can be selected by clicking on it in the Design or wxGlade: Tree window. Note that the names assigned to controls are used in the glue code to reference them. Use care to select names that will help indicate which item is which later. You should not create any bindings for events here; they will be overridden later.
8. When the GUI design is complete, the code can be created using the File/Generate Code menu item. This creates the python file that was referenced in the second step. This file can be run in Python. It will display the frame and controls, but not do anything.
9. Create a template file by running the wxGlue script in Python:

   ```
   python <path>/wxGlue/__init__.py <GUI_file.py>
   ```

   If <GUI_file.py> is not placed on the command line, it is prompted for. The output file name is also prompted for as well.
10. Manually edit the template file to change the way that the controls are used.

CHAPTER

FOUR

4. DESIGNING A WXGLUE PROGRAM

A program using wxGlue will have several sections for each frame; these are outlined below. A template file can be created by wxGlue which will have all of the following sections, as appropriate.

4.1 Imports

The first section will define references for modules that need to be imported:

```
import sys
sys.path.insert(0,"/Users/toby/projects/wxglade")
import wxglue
sys.path.insert(0,"/Users/toby/myProject")
import myGUI
```

In the above case, myGUI.py (in directory /Users/toby/myProject) is the GUI created in wxGlade. The sys.path.insert lines may not be needed or will require editing to make relative to make the resulting code more portable.

4.2 Define Function

The next section defines a function for each wx.Frame and wx.Dialog class found in the the GUI file. It starts by creating a frame from the class and glue object for the frame:

```
def myGUI_MyFrame1(parent=None):
    gluemod = wxglue.wxGlue(myGUI.MyFrame1,parent)
```

Note that a value for parent is not needed when a frame will be independent (non-modal) manner, but is required when a Dialog will be used in a modal fashion. A modal window is one that “freezes the application” until satisfied; the modal window must be closed before other windows can be used. The parent will be the name of the wxglue.wxGlue object of the parent frame.

4.3 Define Callbacks

Then functions are created for each callback that will be invoked by use of controls. Note that almost every control can be linked to a callback, but the template file will be created with examples only for every button, such as this:

```
def On_button_1():
    text_ctrl_1 = gluemod.GetValue('text_ctrl_1')
    datepicker_ctrl_1 = gluemod.GetValue('datepicker_ctrl_1')
    print('Called On_button_1 with values'
          +'\n\ttext_ctrl_1 = '+ str(text_ctrl_1)
          +'\n\tdatepicker_ctrl_1 = '+ str(datepicker_ctrl_1)
          )
```

This code shows how values are obtained for controls (in this case named *text_ctrl_1* and *datepicker_ctrl_1*). These values are then used in the print statement.

In the case of applications that contain both Frame and Dialog objects, then a prototype call to invoke the Dialog as a modal window will also be included in Frame callbacks:

```
# dialog_MyDialog(gluemod)
```

Likewise, in Dialog callbacks, it is assumed that the buttons are intended to close the Dialog frame and a call to do that is included:

```
gluemod.CloseFrame()
```

4.4 Control Initalization

The next section will have sample code for each control found in the frame, for example:

```
# gluemod.SetTextCtrlType("text_ctrl_1", float)
# gluemod.SetValue("text_ctrl_1", 1.0)
# gluemod.SetEmptyInvalid("text_ctrl_1", True)
# gluemod.SetMinValue("text_ctrl_1", -10.0)
# gluemod.SetMaxValue("text_ctrl_1", 10.0)

gluemod.SetAction("button_1", On_button_1)
# gluemod.SetLabel("button_1", 'Button Label')
# gluemod.Enable("button_1", True)

# gluemod.SetValue("datepicker_ctrl_1", (2013,1,31))
```

These examples will show the most commonly used methods for each type of control, but there is a complete list of methods documented for class `wxGlue`. Some methods, such as `wxGlue.Enable()`, are available for every type control. The `wxGlue.SetLabel()` or `wxGlue.SetValue()` methods (occasionally both) will be available for nearly every control, but note that the type for the value associated with the `wxGlue.SetValue()` method will vary, depending on the control type. Note that if a method is not implemented for a control, it will be ignored. Examples of the more commonly used of these statements are used in this section of the code to initialize controls, but they may also be placed in the function callbacks to change values, etc.

4.5 Resize and Show the Frame

Either `wxGlue.ResizeFrame()` or `wxGlue.RedoLayout()` must be called at least once, or the frame is never shown. This causes the frame to be sized for the minimum that will fit all controls:

```
gluemod.ResizeFrame()
```

4.6 Start the code

The final section launches the application by creating the GUI and starting the event loop:

```
myGUI_MyFrame1()
wxglue.StartEventLoop()
```

CHAPTER

FIVE

5. WXGLUE METHODS FOR SPECIFIC CONTROLS

The wxGlue method :func:‘wxGlue.Enable can be used with any control. A number of other methods, such as `wxGlue.SetLabel()`, `wxGlue.SetValue()`, `wxGlue.GetValue‘and :func:‘wxGlue.SetAction()`, can be used with the majority of controls. The sections below describe use cases for particular types of controls.

5.1 wx.TextCtrl methods

One control with special treatment is wx.TextCtrl. This has a two unique methods that are used for validation of the control contents. The method `wxGlue.SetTextCtrlType()` sets a type for the control. At present, `str` (the default), `int` and `float` are all available. For str type controls, `wxGlue.SetEmptyInvalid()` indicates that the TextCtrl control may not be left blank. For the latter two numeric types, only digits, signs and for floats, a decimal point and exponent (e) are allowed to be entered into the control. Also for two numeric types, `wxGlue.SetMinValue()` and `wxGlue.SetMaxValue()` are optionally used to set a range for number that can be entered. If an invalid value is entered into a control, the color of the control is changed to highlight it and other controls may be disabled (see `wxGlue.SetValidationRequired()`).

5.2 wx.Button methods

There are no methods that are special to wx.Button controls, but several methods are used more commonly with buttons than with other controls. Method `wxGlue.SetAction()` is used to specify a function or method that will be called when the button is pressed. Note that optional arguments can specify a list of dictionary of arguments to be supplied in that call. Method `wxGlue.SetValidationRequired()` indicates that a button should be disabled if invalid values are present in any validated TextCtrl. `wxGlue.SetLabel()` can be used to change the label on a button. An OK button (with Id `wx.ID_OK`) is automatically set to be disabled if any input is invalid.

5.3 Calendar Methods

Calendar controls (wx.calendar.CalendarCtrl and wx.DatePickerCtrl) take special input with `wxGlue.SetValue()`. The value can be a list or tuple with three int elements, (year, month, day), such as (2013, 1, 13) or can be a Python datetime.date value. The `wxGlue.GetValue()` routine will always return a datetime.date value.

5.4 Option Selection Methods

Controls that present the user with a set of fixed choices (RadioBox, Choice, ComboxBox, and ListBox) also work a bit differently. The `wxGlue.SetValue()` and `wxGlue.GetValue()` methods accept or return integer values corresponding to the option number (counting starts with zero). Likewise, `wxGlue.GetMinValue()` returns 0 and `wxGlue.GetMaxValue()` returns the number of options minus one. The `wxGlue.SetLabel()` function accepts a list or tuple of string values. For Choice, ComboxBox, and ListBox controls, the number of options will be set by the number of items in the list, while for RadioBox controls, this number cannot be changed and extra strings will be ignored. Note that `wxGlue.GetLabel()` can be used to determine the string value that has been selected in the control.

5.5 StatusBar Methods

The status bar is shown at the bottom of a frame. It supports two methods, `wxGlue.GetLabel()` and `wxGlue.SetLabel()`. The `wxGlue.SetLabel()` method accepts the label for the status bar and a value which can be a sting or a list of strings. If a list of strings, then the status bar is divided into the appropriate number of sections. `wxGlue.GetLabel()` method always returns a list of strings.

5.6 ToolBar methods

The toolbar is a section at the top of the frame where small icon buttons are placed. These buttons can be used to initiate an action, be used as a check button or be used as a radio button. The radio button implementation is not available in wxGlue.

wxPython does not seem to have the ability to discover buttons that are added to the toolbar in the initial GUI code, so these buttons are deleted. Buttons must be added to the toolbar using the `wxGlue.AddItem()` method. As an example, in this code:

```
lbl = gluemod.AddItem("frame_1_toolbar",
                      label="lbl",
                      BitmapFile="/Users/toby/software/wxGlue/glade-tests/text.gif",
                      ToolTip="help txt",HelpText='longer txt')
```

here the initial parameter itentifies the label of the toolbar. The BitmapFile parameter specifies a small (one hopes) icon that can be read from a variety of file formats. The ToolTip provides text that floats over the icon if the mouse is positioned there for a short time. The label and HelpText parameters do seem not to be used.

5.7 MenuBar methods

A wx.MenuBar is placed at the top of a frame (except on a Mac, where it is placed into the system menu bar, usually at the top of the screen). The MenuBar contains a series of wx.Menu items (File, Help, Edit, etc.) and each of these menus contains wx.MenuItems. The MenuItems can hold a value (check menu items and radio menu items), can initiate an action (menu buttons) or can hold a submenu. The `wxGlue.SetAction()` method will be needed to assign a callback to menu buttons, but can also be used in the unusual cases where check menu items and radio menu items need to perform actions other than assigning variables. Note that for all types of MenuItems, help information is shown in the status bar when supplied.

All items in menus will be found and will be assigned glue methods (labels will be concatenations of the frame name and the menu items) but Menus can also be added to a MenuBar with a `wxGlue.AddItem()` method such as:

```
menu = gluemod.AddItem("frame_1_menubar", label="new menu", position=0)
```

note that if the position argument is omitted, the menu is placed at the end of the menu list (usually not what you want).

Menu items can be added to a menu using a `wxGlue.AddItem()` method call, such as:

```
btn = gluemod.AddItem(menu, label="Action",HelpText="Does Action",itemType="button")
def OnAction(): print "OnAction"
gluemod.SetAction(btn, OnAction)
```

Note that itemType can be set to "check" for that type of MenuItems, or to "menu", to create a submenu, as is shown here:

```
def OnAction(val): print "OnAction=",val
submenu = gluemod.AddItem(menu, label="submenu",HelpText="help txt",itemType="menu")
btn = gluemod.AddItem(submenu, label="Action1",HelpText="Does Action 1",itemType="button")
gluemod.SetAction(btn, OnAction,args=[1])
btn = gluemod.AddItem(submenu, label="Action2",HelpText="Does Action 2",itemType="button")
gluemod.SetAction(btn, OnAction,args=[2])
```

5.8 Using a checkbox to enable other controls

A common control motif is that a TextCtrl or other control(s) are enabled or disabled by a checkbox (or sometimes using another control that offers more options as are described in the previous section.) This is easy to do, as is shown in the code example here:

```
def OnCheckbox_1():
    state = gluemod.GetValue("checkbox_1")
    if state:
        gluemod.Enable("text_ctrl_1")
    else:
        gluemod.Disable("text_ctrl_1")
gluemod.SetAction("checkbox_1", OnCheckbox_1)
gluemod.SetValue("checkbox_1", False)
OnCheckbox_1()
```

The Python expert will note that the function above can be rewritten more tersely and less transparently as:

```
def OnCheckbox_1():
    gluemod.Enable("text_ctrl_1",gluemod.GetValue("checkbox_1"))
```

CHAPTER

SIX

6. OTHER IMPORTANT FUNCTIONS AND METHODS

Functions `wxGlue.ResizeFrame()` and `wxGlue.RedoLayout()` are used to arrange (or rearrange) the controls within a frame to account for any change in size. The difference between the two is that `wxGlue.ResizeFrame()` will change the size of the frame, while `wxGlue.RedoLayout()` preserves the current frame size, even if it is not big enough to show all the controls.

For a wxPython application to run, it must enter a state where the program is waiting for the user to do something, such a press a keyboard key, move the mouse, click a mouse button, etc. These user actions are called events. For the application to respond to events, the program must enter a state called the event loop, where the program sleeps, except when an event occurs. This is usually the last thing done in the program and is done by calling `StartEventLoop()`. Note that once this routine is called, it does not return until all open frames are closed.

To close a frame in response to a button, etc. use method `wxGlue.CloseFrame()`. To register a function that will be called when the frame is closed using the close button (X) on the border around the frame, use `wxGlue.SetCloseAction()` in this manner:

```
def On_close():
    print "On Close"
    gluemod.CloseFrame()
gluemod.SetCloseAction(On_close)
```

Likewise, to set a prompt to ask a user to confirm that they want to close a frame (and the application, if it is the only open frame), use this:

```
def On_close():
    if wxglue.GetYesNo('Ok to quit?',gluemod):
        gluemod.CloseFrame()
gluemod.SetCloseAction(On_close)
```

Be sure to use `wxGlue.CloseFrame()` inside the callback, or the frame will not be closed.

A few other methods and functions are worth noting. Routines `GetInputFile()` and `GetNewFile()` are used to open a dialog where a user can select the names of files to be read or written, respectively. Routine `GetString()` opens a dialog for one line of input from the user and `GetYesNo()` opens a dialog for the user to click on Yes or No.

CHAPTER

SEVEN

7. REFERENCE: CLASS AND FUNCTION DOCUMENTATION

`wxGlue.`**`GetInputFile`**(*parentglue=None*, *multiple=False*, *startdir=None*, *title='Choose an input file'*, *wildcard='Python source (*.py)|*.py'*)

Creates a dialog to get name(s) of existing file(s) and returns a list of files

Parameters

- **parentglue** (*frame*) – Name of parent glue object for the dialog or None (default) if not needed
- **multiple** (*bool*) – Allow dialog to select only one file (False, default) or multiple files (True)
- **startdir** (*str*) – string with the name of a directory to initialize the dialog with. If None (default) the initial dialog will be the current working directory.
- **title** (*str*) – a title for the dialog (default is "Choose an input file")
- **wildcard** (*str*) – A string that defines the files that will be searched. The default, "Python source (*.py)*|.py", shows how one is constructed. Here is an example with more than extension: "text file (*.txt)*|.txt|rich text (*.rtf)*|.rtf|any (.)|*.*"

Returns a list of file(s) or an empty list if Cancel is pressed.

`wxGlue.`**`GetNewFile`**(*parentglue=None*, *startdir=None*, *defaultname=''*, *title='Choose an output file'*, *wildcard='Python source (*.py)|*.py'*)

Creates a dialog to get a name for a new file and returns that name

Parameters

- **parentglue** (*frame*) – Name of parent glue object for the dialog or None (default) if not needed
- **startdir** (*str*) – string with the name of a directory to initialize the dialog with. If None (default) the initial dialog will be the current working directory.
- **defaultname** (*str*) – A default file name for the dialog (default is none)
- **title** (*str*) – a title for the dialog (default is "Choose an output file")
- **wildcard** (*str*) – A string that defines the files that will be searched. The default, "Python source (*.py)*|.py", shows how one is constructed. Here is an example with more than extension: "text file (*.txt)*|.txt|rich text (*.rtf)*|.rtf|any (.)|*.*"

Returns a list of file(s) or an empty list if Cancel is pressed.

`wxGlue.`**`GetString`**(*prompt*, *parentglue=None*, *title=''*, *default=''*)

Creates a dialog to get a line of input from the user

Parameters

- **prompt** (*str*) – a string to tell the user what they are inputting
- **parentglue** (*frame*) – Name of parent glue object for the dialog or None (default) if not needed
- **title** (*str*) – a title for the dialog (default is “”)
- **default** (*str*) – a default value to place in the window dialog (default is “”)

Returns a string or None, if Cancel is pressed.

`wxGlue.`**`GetYesNo`**(*prompt*, *parentglue=None*, *title='‘*, *default=True*)

Creates a dialog to get a single Yes/No choice from the user

Parameters

- **prompt** (*str*) – a string to tell the user what they are inputting
- **parentglue** (*frame*) – Name of parent glue object for the dialog or None (default) if not needed
- **title** (*str*) – a title for the dialog (default is “”)
- **default** (*bool*) – a default value for the answer (the default, True, indicates Yes)

Returns False (No) or True (Yes)

`wxGlue.`**`SetupTemplate`**(*in_files=*[ ])

This routine is used to process one or more GUI scripts created in wxGlade and create a template file to help in the creation of a script that uses the GUI script(s) and wxGlue. This will normally be used by invoking this script directly in a python interpreter (python wxGlue/__init__.py).

`wxGlue.`**`StartEventLoop`**()

Start the wx event loop after all code has been put in place

class `wxGlue.`**`wxGlue`**(*frameclass*, *parent=None*)

An object that “glues” controls to a frame (window) object. The frame is created from a class that is typically coded with wxGlade. The name of the class that defines the frame is passed to wxGlue on initialization. Calls to methods are then used to define settings for controls or actions linked to them

Parameters

- **frameclass** (*class*) – defines a class that can be used to create a frame. This class is usually coded using wxGlade.
- **parent** (*frame*) – defines a parent for the current frame. The default is None which indicates this frame has no parent

`AddItem`(*lbl*, *itemType='button'*, *label='‘*, *BitmapFile=None*, *disabledBitmapFile=None*, *ToolTip='‘*, *HelpText='‘*, *position=None*)

Adds an item to a ToolBar or a MenuBar or a Menu.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **itemType** (*str*) – The type of item to be created. For a MenuBar this is ignored; for a ToolBar this can be ‘button’ or ‘check’; for a Menu this can be ‘button’, ‘menu’ or ‘check’. The default is ‘button’.
- **label** (*str*) – a text label for the item. This appears to be ignored for a ToolBar, but is required to be non-blank for everything else.

- **BitmapFile** (*str*) – Name for a file containing a bitmap for the icon. Used for a ToolBar only. If None or the file does not exist, a question mark or exclaimation mark is used. If there is an error reading an image from the file an error is shown.
- **disabledBitmapFile** (*str*) – Name for a file containing a bitmap for the icon when the ToolBar button is disabled (ToolBar only). If None there is no change in the icon when disabled.
- **ToolTip** (*str*) – Text to be displayed as a ToolTip for ToolBar. This is text that floats over the button if the mouse is allowed to linger over that button. (Default is none). Used for ToolBar only.
- **HelpText** (*str*) – Text displayed in the StatusBar (if present) for MenuItems. Does not seem to be used for ToolBar buttons. (Default is none).
- **position** (*int*) – Position for where item should be placed. If 0 it is placed at the beginning. If None (default) it is placed at the end.

Returns the name of the newly created control

`CloseFrame()`
Closes the frame (window) created by this object.

`Disable(`*lbl*`)`
Disables a control.

Parameters **lbl** (*str*) – the name of the control (widget)

`Enable(`*lbl*, *value=True*`)`
Enables or Disables a control. This can be used with all controls, but ones that do not have a used interaction may not change as a result of being disabled.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*bool*) – Determines if the control will be enabled (value=True, default) or disabled (value=False).

`GetLabel(`*lbl*`)`
Returns the label associated with a control or a list of labels in the case of controls that select options. Note that some controls do not display a label, but still can have a label set. Controls that contain multiple options (wx.RadioBox, wx.Choice, wx.ComboBox or wx.ListBox) return a list of labels (each an str).

Parameters **lbl** (*str*) – the name of the control (widget)

Returns the widget label, if defined (type str) or a list of (str) labels for the wx.RadioBox, wx.Choice, wx.ComboBox or wx.ListBox controls.

`GetMaxValue(`*lbl*`)`
Get a maximum value that is allowed for the setting of a control. This can be used for TextCtrl, SpinButton and SpinCtrl controls as well as controls that have options (RadioBox, Choice, ComboxBox, and ListBox) where it will supply the maximum allowed value.

Parameters **lbl** (*str*) – the name of the control (widget)

Returns the maximum allowed value for control *lbl* (int/float)

`GetMinValue(`*lbl*`)`
Get a minimum value that is allowed for the setting of a control. This can be used for TextCtrl, SpinButton and SpinCtrl controls as well as controls that have options (where it will supply the minimum valid value, 0).

Parameters **lbl** (*str*) – the name of the control (widget)

Returns the minimum allowed value for control *lbl* (int/float)

`GetValue`(*lbl*)

Returns the value associated with a control. The type for the value is dictated by the type of the control, as shown in the table below:

Parameters **lbl** (*str*) – the name of the control (widget)

control	type
wx.TextCtrl	str or [float/int, if set by `SetTextCtrlType()`]
wx.DatePickerCtrl	*datetime.date*
wx.calendar.CalendarCtrl	*datetime.date*
wx.ToggleButton	bool
wx.SpinCtrl	int
wx.CheckBox	bool
wx.RadioButton	bool
wx.RadioBox	int
wx.Gauge	float
wx.Slider	int
wx.Choice	int
wx.ComboBox	int
wx.ListBox	int
wx.Button	None
wx.StaticBitmap	None
wx.StaticText	None
wx.StaticBox	None

`RedoLayout`(*hide=False*)

Redo the layout of the frame (window) created by this object to account for changes in labels, etc. This does not change the size of the frame.

Parameters **hide** (*bool*) – If True, this causes the frame to be hidden from view. The default (False) causes the frame to be shown.

`ResizeFrame`(*width=-1*, *height=-1*, *hide=False*)

Resizes the frame (window) created by this object to account for changes in labels, etc. Optionally a size for the frame can be specified here.

Parameters

- **width** (*int*) – This specifies the width for the frame in pixels. The default (-1) is to use the minimum needed to display all objects.
- **height** (*int*) – This specifies the height for the frame in pixels. The default (-1) is to use the minimum needed to display all objects.
- **hide** (*bool*) – If True, this causes the frame to be hidden from view. The default (False) causes the frame to be shown.

`SetAction`(*lbl*, *routine*, *args*=[ ], *kwargs={}*)

Register a function that will be called when a control is activated. This will work with almost every type of input control, but will most commonly be used with button-type controls. For controls where input can be typed (TextCtrl and SpinBox, for example), these actions will be performed every time then mouse leaves the text box, so any action routine should be innocuous.

Parameters

- **lbl** (*str*) – the name of the control (widget)

- **routine** (*function*) – a function or method that will be called when the control is invoked, for example, by pressing a button
- **args** (*list*) – optional positional arguments to be supplied as parameters to the function *routine*
- **kwargs** (*dict*) – optional keyword arguments to be supplied as parameters to the function *routine*

`SetCloseAction`(*routine*, *args=[ ]*, *kwargs={}*)

Register a function that will be called when a frame is closed. Note that no arguments to that routine can be passed.

Parameters

- **routine** (*function*) – a function or method that will be called when the control is invoked, for example, by pressing a button
- **args** (*list*) – optional positional arguments to be supplied as parameters to the function *routine*
- **kwargs** (*dict*) – optional keyword arguments to be supplied as parameters to the function *routine*

`SetEmptyInvalid`(*lbl*, *value=True*)

Indicate that a TextCtrl control is invalid if it has a blank value. A control is considered to be blank if it has only whitespace characters [see str.strip()] or is null. The default is to allow controls to be considered valid if blank.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*bool*) – Value to set: True (defaults) indicates that a control is invalid when blank

`SetLabel`(*lbl*, *value*)

Sets the label(s) associated with a control. This can be used with all controls, but for controls do not display a label, for these items no change will be seen. Controls that contain multiple options (wx.RadioBox, wx.Choice, wx.ComboBox or wx.ListBox) require a list of labels (each an str). Note that for wx.RadioBox, the number of options cannot be changed, but for the rest, the number of labels in the list determines the new number of options in the control.

If labels change in length, it may be necessary to change the layout of the frame, or change the frame size as well; this is done with `ResizeFrame()` and `RedoLayout()`, respectively.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*str/list*) – the new label for control *lbl*, or a list of labels for wx.RadioBox, wx.Choice, wx.ComboBox or wx.ListBox.

`SetMaxValue`(*lbl*, *value*)

Set a maximum value that is allowed for the setting of a control. This can be used for SpinButton and SpinCtrl controls as well as TextCtrls.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*int/float*) – the maximum allowed value for control *lbl*

`SetMinValue`(*lbl*, *value*)

Set a minimum value that is allowed for the setting of a control. This can be used for SpinButton and SpinCtrl controls as well as TextCtrls.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*int/float*) – the minimum allowed value for control *lbl*

`SetTextCtrlType`(*lbl*, *typ*)

Set a variable type for a TextCtrl control. If the type is int or float, restrictions are placed on the characters that may be typed in the control. For int, valid characters are "+-0123456789", for float, the characters ".", "e" and "E" are allowed in addition. Validation of the contents of the TextCtrl is performed when Return is pressed or another control is selected. This means checking if the can be converted to a valid int or float, if selected, or or is non-blank, if `SetEmptyInvalid()` has been used to require non-blank values.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **typ** (*type*) – a type to be used for the TextCtrl value. Usually int or float (str is assumed if not called).

`SetValidationRequired`(*lbl*, *value=True*)

Indicate that a control (usually Button) will be disabled when validated TextCtrl controls are set with invalid values. Do not use `SetValidationRequired()` with TextCtrls because once the control has an invalid value, it will be disabled and then can't be changed.

Note that OK buttons (with Id = wx.ID_OK) are automatically set to be disabled when validated TextCtrl controls are set with invalid values.

TextCtrl controls are set to be validated using `SetTextCtrlType()`, `SetMinValue()` and `SetMinValue()` and/or `SetEmptyInvalid()`.

Parameters

- **lbl** (*str*) – the name of the control (widget)
- **value** (*bool*) – Value to set: True (default) indicates that a control should be disabled when validated controls are invalid.

`SetValue`(*lbl*, *value*)

Change the value associated with a control. Note that the type for the value must be appropriate for the control, as shown in the table below.

Parameters **lbl** (*str*) – the name of the control (widget) :param many value: the value for control *lbl*, with type dictated by the table below. Note that for controls labeled (n/a), this method has no effect.

control	type
wx.TextCtrl	str or [float/int, if set by `SetTextCtrlType()`]
wx.DatePickerCtrl	*datetime.date* or list of 3 integers as (yyyy,mm,dd)
wx.calendar.CalendarCtrl	*datetime.date* or list of 3 integers as (yyyy,mm,dd)
wx.ToggleButton	bool
wx.SpinCtrl	int
wx.CheckBox	bool
wx.RadioButton	bool
wx.RadioBox	int
wx.Gauge	float
wx.Slider	int
wx.Choice	int
wx.ComboBox	int
wx.ListBox	int
wx.Button	(n/a)
wx.StaticBitmap	(n/a)
wx.StaticText	(n/a)
wx.StaticBox	(n/a)

`ShowValues``()`
For development, print all values in the self.Values dict.

`TestIfValid``()`
Check if all TextCtrl controls that have defined ranges or require non-empty values have been set with valid values. All controls that designated to require validation [with self.SetValidationRequired()] will be enabled or disabled accordingly. This will probably not be used externally.

Returns True if all validated TextCtrl controls have valid values

`Values` **= None**
A dict indexed by a control label. Contains the current value for that input control. Also see `GetValue()`, which returns a value from this dict.

PYTHON MODULE INDEX

W

wxGlue, 1

INDEX

A

AddItem() (wxGlue.wxGlue method), 20

C

CloseFrame() (wxGlue.wxGlue method), 21

D

Disable() (wxGlue.wxGlue method), 21

E

Enable() (wxGlue.wxGlue method), 21

G

GetInputFile() (in module wxGlue), 19
GetLabel() (wxGlue.wxGlue method), 21
GetMaxValue() (wxGlue.wxGlue method), 21
GetMinValue() (wxGlue.wxGlue method), 21
GetNewFile() (in module wxGlue), 19
GetString() (in module wxGlue), 19
GetValue() (wxGlue.wxGlue method), 22
GetYesNo() (in module wxGlue), 20

R

RedoLayout() (wxGlue.wxGlue method), 22
ResizeFrame() (wxGlue.wxGlue method), 22

S

SetAction() (wxGlue.wxGlue method), 22
SetCloseAction() (wxGlue.wxGlue method), 23
SetEmptyInvalid() (wxGlue.wxGlue method), 23
SetLabel() (wxGlue.wxGlue method), 23
SetMaxValue() (wxGlue.wxGlue method), 23
SetMinValue() (wxGlue.wxGlue method), 23
SetTextCtrlType() (wxGlue.wxGlue method), 24
SetupTemplate() (in module wxGlue), 20
SetValidationRequired() (wxGlue.wxGlue method), 24
SetValue() (wxGlue.wxGlue method), 24
ShowValues() (wxGlue.wxGlue method), 25
StartEventLoop() (in module wxGlue), 20

T

TestIfValid() (wxGlue.wxGlue method), 25

V

Values (wxGlue.wxGlue attribute), 25

W

wxGlue (class in wxGlue), 20
wxGlue (module), 1